RPKI Operations

ThaiNOG 4

24th May 2022

Carlton Hotel | Bangkok Sukhumvit

UNIVERSITY OF OREGON

Last updated 22nd May 2022

NSRC
Network Startup Resource Center

Background

- Most operators know to sign their ROAs (Route Origin Authorisations)
  - Major push by all five Regional Internet Registries
  - Major awareness effort by MANRS (https://www.manrs.org)
  - Efforts by many across the Internet operations community
- But what about making use of these ROAs?
  - What needs to be done?
  - How does a network operator deploy?
  - Which operators need to deploy?

Mutually Agreed Norms for Routing Security

- Implement the MANRS recommendations:
  - Prevent propagation of incorrect routing information
    - Filter BGP peers, in & out!
  - Prevent traffic with spoofed source addresses
    - BCP38 – Unicast Reverse Path Forwarding
  - Facilitate communication between network operators
    - NOC to NOC Communication
    - Up-to-date details in Route and AS Objects, and PeeringDB
  - Facilitate validation of routing information
    - Route Origin Authorisation using RPKI

RPKI Operations

- The three steps towards using RPKI every day:
  1. Deploy validators
  2. EBGP speaking routers talk with validators
  3. Making decisions about dropping invalid routes
- Each step is a significant change in daily operations
  - Each one should be done in turn…
  - With reviews and monitoring to gain operational experience…
  - And not moving to next step until satisfied…
- There are gotchas too!

Route Origin Authorisation

- A typical ROA would look like this:

Prefix	10.10.0.0/16
Max-Length	/18
Origin-AS	AS65534

- There can be more than one ROA per address block
  - Allows the operator to originate prefixes from more than one AS
  - Caters for changes in routing policy or prefix origin

Step 1

VALIDATORS

UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

RPKI Validator Caches (1)

- NLnet Labs Routinator 3000
  - https://www.nlnetlabs.nl/projects/rpki/routinator/
  - https://github.com/NLnetLabs/routinator
  - Packages available for Debian/Ubuntu, RHEL/CentOS & FreeBSD
  - (Can also be built from source)

- LACNIC/NIC Mexico validator (FORT)
  - https://fortproject.net/en/validator
  - https://nicmx.github.io/FORT-validator/
  - Packages available for Debian/Ubuntu, RHEL/CentOS & FreeBSD
  - (Can also be built from source)

RPKI Validator Caches (2)

- RPKI-client
  - https://www.rpki-client.org/
  - https://tracker.debian.org/pkg/rpki-client
  - RPKI repository query system (output for OpenBGPD, BIRD, json)
  - For OpenBSD, with ports for Debian/Ubuntu, RHEL/CentOS, FreeBSD, macOS
- StayRTR
  - https://github.com/bgp/stayrtr
  - https://tracker.debian.org/pkg/stayrtr
  - RPKI to Router protocol implementation (input JSON formatted VRP exports)
  - (hard fork of Cloudflare GoRTR)
  - Works on anything Go runs on (?)
- Note:
  - RPKI-client and StayRTR are used together

RPKI Validator Caches (3)

- RPKI-Prover
  - https://github.com/lolepezy/rpki-prover
- rpstir2
  - https://github.com/bgpsecurity/rpstir2

- The following are no longer maintained – please don't use them!
  - Dragon Research Labs "rcynic"
  - Cloudflare validator (OctoRPKI/GoRTR)
    - StayRTR is a fork of GoRTR
  - RIPE NCC validator

Installing a validator

- Three validators are widely used
  - Routinator
  - FORT
  - RPKI-client/StayRTR
- Listed in order of ease of installation 😀
- For installation details on Ubuntu 20.04
  - https://bgp4all.com/pfs/hints/rpki

Validator Deployment

- Deploy at least two Validator Caches
- Geographically diverse
- At least two different implementations
  - For software independence
  - Standards interpretation
- Implement each on a Linux container so that the container can be moved around as required
- Configure validator to listen on both IPv4 and IPv6
  - Configure routers with both IPv4 and IPv6 validator connections
- Securing the validator: Only permit routers running EBGP to have access to the validators

Monitor the Validator

- To get an understanding of what is going on, monitor the validators:
  - What does the validation cache look like?
    - Routinator has a web interface to let you see the cache
    - RPKI-client's JSON output?
  - What is their start-up time like?
    - Routinator & FORT sync the caches each time the process starts – so it can be 15-20 minutes before they are ready to serve data to any router
    - RPKI-client and StayRTR are independent processes – and StayRTR is ready as soon as it is started, using the latest dataset built by RPKI-client
  - What are the memory, CPU, and physical storage resources like?
    - Validation data currently requires about 2.2Gbytes of storage (and growing)

Step 2

VALIDATOR TO ROUTER

UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

Connecting Validators to Routers

- Significant step, as this is touching the operating network
  - Deploying a validator had no operational impact

- Only configure EBGP speaking routers to talk to the validators
  - Routers receive VRPs (Validated ROA Payloads)
- Nothing to be gained by:
  - Configuring IBGP speakers to talk to validators
    - Route Origin Validation (Step 3) is done at the edge
    - The core never needs to know, invalids not sent there!
  - Propagating validation information through IBGP

Connecting Validators to Routers

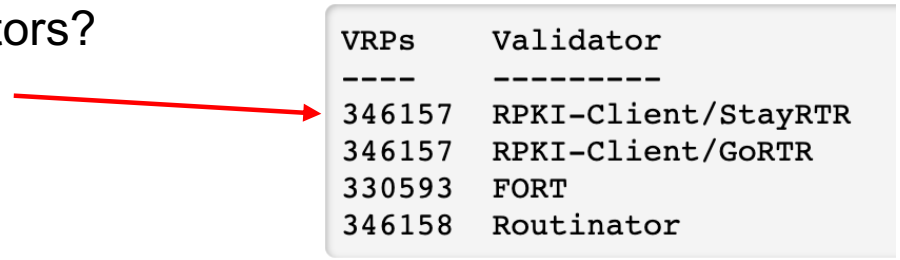
- Be very aware of vendor default behaviour!!
- The ideal behaviour is:
  - Router creates internal validation database
  - Operator configures policy that flags what is Valid, Invalid, and NotFound in the BGP RIB
  - Operator configured policy that determines if invalid routes are dropped or propagated
- Cisco IOS-XE/XR has very different behaviour from JunOS, BIRD, and FRrouting
  - https://bgp4all.com/pfs/hints/rpki
- What does your implementation do?

Connecting Validators to Routers

- How does your implementation react to changes in validation info?
  - For example: route changes from invalid to valid or notfound
  - Does it send a Route Refresh to peers?
  - Or does it maintain an ADJ-RIB-IN?
    - BGP table separate from the active BGP RIB
    - Cisco IOS “soft-reconfiguration” knob is similar
- It’s important not to rely on Route Refresh to implement VRP changes
  - More and more frequent changes cause more and more refresh requests to peers, consuming peer CPU resources – potentially a denial of service attack on the peer
  - Recommended reading:
    - https://datatracker.ietf.org/doc/draft-ymbk-sidrops-rov-no-rr/

Connecting Validators to Routers

- Considerations:
  - Can you work around the vendor defaults to get the behaviour you want just to monitor what is valid/invalid/notfound
  - How many validators do you need (minimum of 2 recommended)
  - How to deal with validator start-up time (Routinator & FORT)
  - What about VRP differences between validators?
    - https://bgp.nsrc.org/REN/rpki/validator.state.html
- Monitoring phase:
  - Check the validation database on the router
  - Look for prefixes in BGP table marked as invalid
    - (But don't throw anything away)
  - Consider potential customer impact?



```
VRPs    Validator
----    ---------
346157  RPKI-Client/StayRTR
346157  RPKI-Client/GoRTR
330593  FORT
346158  Routinator
```

Step 3

ROUTE ORIGIN VALIDATION

UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

Route Origin Validation

- Final deployment step: turn on ROV!
  - Treat the same way as any major BGP policy change – planned maintenance!

- Where first?
  - BGP Customers
  - Private Peers
  - Public Peers (IXP)
  - Transit Providers (Upstreams)

  - Or all EBGP peers regardless?

Route Origin Validation

- And plan what needs to be done on the routers
- For Cisco IOS-XE
  - Remove `bgp bestpath prefix-validate allow-invalid` on all EBGP speaking routers
- For Juniper/FRR/BIRD/…
  - Implement policy to drop invalids
    - Inbound on EBGP peer, but then operator will never see invalids on edge routers
    - Outbound on all BGP peers, so that invalids are never propagated
- For more deployment details & hints:
  - https://bgp4all.com/pfs/hints/rpki

Route Origin Validation

- Information!
  - NOC & Customer Support needs to be fully aware
- Monitor!
  - Any changes in paths used or traffic loads on external links?
  - How to handle these changes?
- End-users may observe connectivity or path change issues
  - Prefixes marked as Invalid because the origin ROA was not set up properly (e.g. aggregate has ROA, subnet does not)

Concluding thoughts

- Who needs to do this?
  - Those interested in helping prevent the propagation of invalid routing information
- Transit Providers:
  - Live in the default free zone (usually)
  - Provide transit to other ASes → Need to do ROV
- IXPs:
  - The IX Route Servers → Need to do ROV
- CDNs & Cloud Providers:
  - Connect to large numbers of peers → Recommended to do ROV
- Access Providers:
  - Are usually multihomed to two or more upstreams, and have default routes
  - No need to do ROV (because of default) but may want to monitor invalid prefixes they receive

Summary

- The three operational steps need to fully deploy RPKI based Route Origin Validation

  1. Setting up validators
  2. Routers talking to validators
  3. Router Origin Validation

- And some of the operational considerations around these